Mackenzie Tjogas
402

HW #1

1.) Stephens 1.1-What are the basic tasks that all software engineering projects must handle?

Requirements gathering, high-level design, low-level design, development, testing, deployment, maintenance, wrap-up

2.) Stephens 1.2-Give a one sentence description of each of the tasks listed in exercise 1

Requirements Gathering-Gathering the customer's wants/needs
High-level Design-Explain the major parts of the application and how they interact together
Low-level Design-More detailed than high-level, explain how to build the application's parts in a way that programmers can use to implement
Development-Write the code to make the application
Testing-Use the application and make sure there are no bugs, despite the edge cases you give it
Deployment-Make the application live to users
Maintenance-Keep the application up and running, fix bugs that are found, add to the application and deploy new versions
Wrap-Up-Look at the project as an overview and see what was done well and what needs to be improved for next time

3.) Stephens 2.4-Like Microsoft Word, Google Docs provides some simple change tracking tools. Go to http://www.google.com/docs/about/ to learn more and sign up [if you do not have an account already]. Then create a document, save it, close it, reopen it, and make changes to it as you did in Exercise 1.

Done.

4.) Stephens 2.5-What does JBGE stand for and what does it mean?

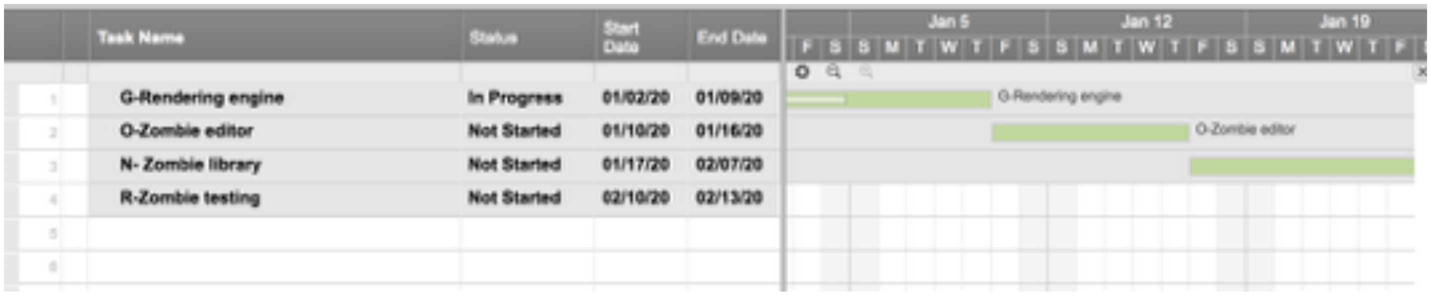
JBGE is a school of thought in software engineering that says you should provide code documentation and comments that are "Just Barely Good Enough", meaning that if you provide too much documentation it will end up wasting your time as you update it for each change made in the code.

5.) Stephens 3.2-Use critical path methods to find the total expected time from the project's start for each task's completion. Find the critical path. What are the tasks on the critical path? What is the total expected duration of the project in working days?

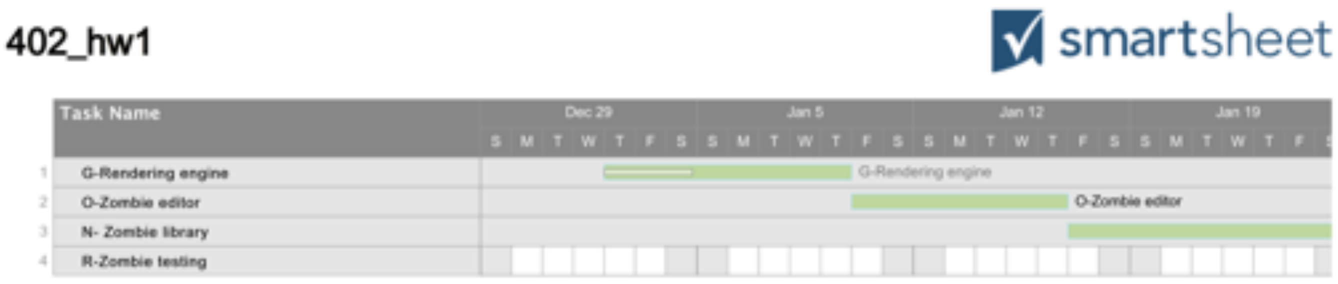
The critical path would be: G —> D —> E —> M —> Q. The tasks for this path are: G-Rendering engine, D-Character editor, E-Character animator, M-Character Library, and Q-Character testing. The total time in working days for this project would be 32 days.

6.) Stephens 3.4-Build a Gantt chart for the network you drew in Exercise 3. [Yes, I know, you weren't assigned that one — however, when you do Exercise 2 you should have enough information for this one.] Start on Wednesday, January 1, 2020, and don't work on weekends or the following holidays:

There are two second-shortest paths (from exercises 3): G —> O —> N —> R and H —> J —> O —> N —> R and both of them total 30 working days.
I did the Gantt chart for the G —> O —> N —> R path.



	Task Name	Status	Start Date	End Date
1	G-Rendering engine	In Progress	01/02/20	01/09/20
2	O-Zombie editor	Not Started	01/10/20	01/16/20
3	N- Zombie library	Not Started	01/17/20	02/07/20
4	R-Zombie testing	Not Started	02/10/20	02/13/20
5				
6				



402_hw1

	Task Name
1	G-Rendering engine
2	O-Zombie editor
3	N- Zombie library
4	R-Zombie testing

7.) Stephens 3.6-In addition to losing time from vacation and sick leave, projects can suffer from problems that just strike out of nowhere. Sort of a bad version of deus ex machina. For example, senior management could decide to switch your target platform from Windows desktop PSs to the latest smartwatch technology. Or a strike in the Far East could delay the shipment of your new servers. Or one of your developers might move to Iceland. How can you handle these sorts of completely unpredictable problems?

To account for unexpected problems you can add tasks to the end of a schedule. If one of these problems does strike up you can insert the lost time into the schedule.

8.) Stephens 3.8-What are the two biggest mistakes you can make while tracking tasks?

The first mistake would be not taking any action when a task slips. One would at least need to watch that task closely so you can act if it's in bad shape. The second mistake would be adding more people to a task in the hopes of shaving time off the total time. Teaching new people and getting them to be on the same page as everyone else can make a task take even longer.

9.) Stephens 4.1-List five characteristics of good requirements.

Good requirements need to be:
1. Easy to understand (clear)
2. Unambiguous
3. Consistent
4.Prioritized
5. Verifiable

10.) Stephens 4.3-Suppose you want to build a program called TimeShifter to upload and download files at scheduled times while you're on vacation. The following list shows some of the applications requirements. List the audience-oriented categories for each requirement. Are there requirements in each category? [If not, state why not…]

a. Allow users to monitor uploads/downloads while away from the office.
    Business
b. Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password.
    User, Functional
c. Let the user specify upload/download parameters such a number of retries if there's a problem.
    User, Functional
d. Let the user select an Internet location, a local file, and a time to perform the upload/ download.
    User, Functional
e. Let the user schedule uploads/downloads at any time.
    Nonfunctional
f. Allow uploads/downloads to run at any time.
    Nonfunctional
g. Make uploads/downloads transfer at least 8 Mbps.
    Nonfunctional
h. Run uploads/downloads sequentially. Two cannot run at the same time.
    Nonfunctional
i. If an upload/download is scheduled for a time when another is in progress, it waits until the other one finishes.
    Nonfunctional
j. Perform schedule uploads/downloads.
    Functional

k. Keep a log of all attempted uploads/downloads and whether the succeeded.
    Functional
l. Let the user empty the log.
    User, Functional
m. Display reports of upoad/download attempts.
    User, Functional
n. Let the user view the log reports on a remote device such as a phone.
    User, Functional
o. Send an e-mail to an administrator if an upload/download fails more than its maximum retry number of times.
    User, Functional
p. Send a text message to an administrator if an upload/download fails more than it's maximum retry number of times.
    User, Functional

All of the audience-oriented categories include at least one requirement.

11.) Stephens 4.9-Brainstorm this application (given online) and see if you can think of ways you might change it. Use the MOSCOW method to prioritize your changes.

M-Must, S-Should, C-Could, W-Won't
Possible changes:
S-should keep score of your "Mr.Bones" record (number of Hangman games you've won/lost).
C-autofill, allow the user to guess the entire word if they think they know it, without having to go key by key.
C-Allow the "hangman" character to be changeable so it doesn't always have to be Mr.Bones
C-Pick the difficulty of the word, a way to incorporate quasi levels in the game. Ex: User could choose if he/she wants a "hard" word or "medium" word, or an "easy" word.
W-don't impose time limits on the user